



OpenCore

Reference Manual (0.5.~~3~~.4)

[2020.01.12]

Failsafe: false

Description: Reuse original hibernate memory map.

This option forces XNU kernel to ignore newly supplied memory map and assume that it did not change after waking from hibernation. This behaviour is required to work by Windows, which mandates to preserve runtime memory size and location after S4 wake.

Note: This ~~option is deprecated and will be removed in the newer OpenCore releases. There is no known hardware that needs this option, and its entire existence appears to be a programmer error. Please report on if you need this option~~ may be used to workaround buggy memory maps on older hardware, and is now considered rare legacy. Examples of such hardware are Ivy Bridge laptops with Insyde firmware, like Acer V3-571G. Do not use this unless you fully understand the consequences.

6. **EnableSafeModeSlide**

Type: plist boolean

Failsafe: false

Description: Patch bootloader to have KASLR enabled in safe mode.

This option is relevant to the users that have issues booting to safe mode (e.g. by holding `shift` or using `-x` boot argument). By default safe mode forces 0 slide as if the system was launched with `slide=0` boot argument. This quirk tries to patch `boot.efi` to lift that limitation and let some other value (from 1 to 255) be used. This quirk requires `ProvideCustomSlide` to be enabled.

Note: The necessity of this quirk is determined by safe mode availability. If booting to safe mode fails, this option can be tried to be enabled.

7. **EnableWriteUnprotector**

Type: plist boolean

Failsafe: false

Description: Permit write access to UEFI runtime services code.

This option bypasses `RX` permissions in code pages of UEFI runtime services by removing write protection (WP) bit from `CR0` register during their execution. This quirk requires `OC_FIRMWARE_RUNTIME` protocol implemented in `FwRuntimeServices.efi`.

Note: The necessity of this quirk is determined by early boot crashes of the firmware.

8. **ForceExitBootServices**

Type: plist boolean

Failsafe: false

Description: Retry `ExitBootServices` with new memory map on failure.

Try to ensure that `ExitBootServices` call succeeds even with outdated `MemoryMap` key argument by obtaining current memory map and retrying `ExitBootServices` call.

Note: The necessity of this quirk is determined by early boot crashes of the firmware. Do not use this unless you fully understand the consequences.

9. **ProtectCsmRegion**

Type: plist boolean

Failsafe: false

Description: Protect CSM region areas from relocation.

Ensure that CSM memory regions are marked as ACPI NVS to prevent `boot.efi` or XNU from relocating or using them.

Note: The necessity of this quirk is determined by artifacts and sleep wake issues. As `AvoidRuntimeDefrag` resolves a similar problem, no known firmwares should need this quirk. Do not use this unless you fully understand the consequences.

10. **ProvideCustomSlide**

Type: plist boolean

Failsafe: false

Description: Provide custom KASLR slide on low memory.

This option performs memory map analysis of your firmware and checks whether all slides (from 1 to 255) can be used. As `boot.efi` generates this value randomly with `rdrand` or pseudo randomly `rdtsc`, there is a chance of boot failure when it chooses a conflicting slide. In case potential conflicts exist, this option forces macOS to use a pseudo random value among the available ones. This also ensures that `slide=` argument is never passed to the operating system for security reasons.

Note: The necessity of this quirk is determined by OCABC: `Only N/256 slide values are usable!` message in the debug log. If the message is present, this option is to be enabled.

11. `SetupVirtualMap`

Type: `plist boolean`

Failsafe: `false`

Description: Setup virtual memory at `SetVirtualAddresses`.

Select firmwares access memory by virtual addresses after `SetVirtualAddresses` call, which results in early boot crashes. This quirk workarounds the problem by performing early boot identity mapping of assigned virtual addresses to physical memory.

Note: The necessity of this quirk is determined by early boot failures.

12. `ShrinkMemoryMap`

Type: `plist boolean`

Failsafe: `false`

Description: Attempt to join similar memory map entries.

Select firmwares have very large memory maps, which do not fit Apple kernel, permitting up to 64 slots for runtime memory. This quirk attempts to unify contiguous slots of similar types to prevent boot failures.

Note: The necessity of this quirk is determined by early boot failures. It is rare to need this quirk on Haswell or newer. Do not use unless you fully understand the consequences.

13. `SignalAppleOS`

Type: `plist boolean`

Failsafe: `false`

Description: Report macOS being loaded through OS Info for any OS.

This quirk is useful on Mac firmwares, which behave differently in different OS. For example, it is supposed to enable Intel GPU in Windows and Linux in some dual-GPU MacBook models.

4. MaxKernel
Type: plist string
Failsafe: Empty string
Description: Blocks kernel driver on specified macOS version or older.
Note: Refer to Add MaxKernel description for matching logic.

5. MinKernel
Type: plist string
Failsafe: Empty string
Description: Blocks kernel driver on specified macOS version or newer.
Note: Refer to Add MaxKernel description for matching logic.

7.5 Emulate Properties

1. Cpuid1Data
Type: plist data, 16 bytes
Failsafe: All zero
Description: Sequence of EAX, EBX, ECX, EDX values ~~in Little Endian order~~ to replace CPUID (1) call in XNU kernel.

This property serves for two needs:

- Enabling support of an unsupported CPU model.
- Enabling XCPM support for an unsupported CPU variant.

Normally it is only the value of EAX that needs to be taken care of, ~~which represents the exact CPUID. And the remainders since it represents the full CPUID. The remaining bytes~~ are to be left as zeroes. ~~For instance, Byte order is Little Endian, so for example, A9 06 03 00 stands for CPUID 0x0306A9 (Ivy Bridge). A good example~~

For XCPM support it is recommended to use the following combinations.

- Haswell-E (0x306F2) to Haswell (0x0306C3):
Cpuid1Data: C3 06 03 00 00 00 00 00 00 00 00 00 00 00 00 00
Cpuid1Mask: FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00
- Broadwell-E (0x0406F1) to Broadwell (0x0306D4):
Cpuid1Data: D4 06 03 00 00 00 00 00 00 00 00 00 00 00 00 00
Cpuid1Mask: FF FF FF FF 00 00 00 00 00 00 00 00 00 00 00 00

Further explanations can be found at acidanthera/bugtracker#365. ~~(See See Special NOTES for Haswell for Haswell+ low-end)~~.

2. Cpuid1Mask
Type: plist data, 16 bytes
Failsafe: All zero
Description: Bit mask of active bits in Cpuid1Data.

When each Cpuid1Mask bit is set to 0, the original CPU bit is used, otherwise set bits take the value of Cpuid1Data.

7.6 Patch Properties

1. Base
Type: plist string
Failsafe: Empty string
Description: Selects symbol-matched base for patch lookup (or immediate replacement) by obtaining the address of provided symbol name. Can be set to empty string to be ignored.
2. Comment
Type: plist string
Failsafe: Empty string
Description: Arbitrary ASCII string used to provide human readable reference for the entry. It is implementation defined whether this value is used.

7.7 Quirks Properties

1. AppleCpuPmCfgLock

Type: plist boolean

Failsafe: false

Description: Disables PKG_CST_CONFIG_CONTROL (0xE2) MSR modification in AppleIntelCPUPowerManagement.kext, commonly causing early kernel panic, when it is locked from writing.

~~Note: This option should~~ Certain firmwares lock PKG_CST_CONFIG_CONTROL MSR register. To check its state one can use bundled VerifyMsR2 tool. Select firmwares have this register locked on some cores only.

As modern firmwares provide CFG Lock setting, which allows configuring PKG_CST_CONFIG_CONTROL MSR register lock, this option should be avoided whenever possible. ~~Modern firmwares provide~~ For several APTIO firmwares not displaying CFG Lock setting, ~~disabling which is much cleaner. More details about the issue can be found in notes~~ in the GUI it is possible to access the option directly:

- (a) Download UEFITool and IFR-Extractor.
- (b) Open your firmware image in UEFITool and find CFG Lock unicode string. If it is not present, your firmware may not have this option and you should stop.
- (c) Extract the Setup.bin PE32 Image Section (the one UEFITool found) through Extract Body menu option.
- (d) Run IFR-Extractor on the extracted file (e.g. ./ifrextract Setup.bin Setup.txt).
- (e) Find CFG Lock, VarStoreInfo (VarOffset/VarName): in Setup.txt and remember the offset right after it (e.g. 0x123).
- (f) Download and run Modified GRUB Shell compiled by brainsucker or use a newer version by datasone.
- (g) Enter setup_var 0x123 0x00 command, where 0x123 should be replaced by your actual offset, and reboot.

WARNING: Variable offsets are unique not only to each motherboard but even to its firmware version. Never ever try to use an offset without checking.

2. AppleXcpmCfgLock

Type: plist boolean

Failsafe: false

Description: Disables PKG_CST_CONFIG_CONTROL (0xE2) MSR modification in XNU kernel, commonly causing early kernel panic, when it is locked from writing (XCPM power management).

Note: This option should be avoided whenever possible. ~~Modern firmwares provide~~ See CFG Lock AppleCpuPmCfgLock setting, ~~disabling which is much cleaner. More details about the issue can be found in notes~~ description for more details.

3. AppleXcpmExtraMsrs

Type: plist boolean

Failsafe: false

Description: Disables multiple MSR access critical for select CPUs, which have no native XCPM support.

This is normally used in conjunction with Emulate section on Haswell-E, Broadwell-E, Skylake-X, and similar CPUs. More details on the XCPM patches are outlined in acidanthera/bugtracker#365.

Note: Additional not provided patches will be required for Ivy Bridge or Pentium CPUs. It is recommended to use AppleIntelCpuPowerManagement.kext for the former.

4. AppleXcpmForceBoost

Type: plist boolean

Failsafe: false

Description: Forces maximum performance in XCPM mode.

This patch writes 0xFF00 to MSR_IA32_PERF_CONTROL (0x199), effectively setting maximum multiplier for all the time.

Note: While this may increase the performance, this patch is strongly discouraged on all systems but those explicitly dedicated to scientific or media calculations. In general only certain Xeon models benefit from the patch.

5. CustomSMBIOSGuid
Type: plist boolean
Failsafe: false
Description: Performs GUID patching for UpdateSMBIOSMode Custom mode. Usually relevant for Dell laptops.
6. DisableIoMapper
Type: plist boolean
Failsafe: false
Description: Disables IOMapper support in XNU (VT-d), which may conflict with the firmware implementation.
Note: This option is a preferred alternative to dropping DMAR ACPI table and disabling VT-d in firmware preferences, which does not break VT-d support in other systems in case they need it.
7. ExternalDiskIcons
Type: plist boolean
Failsafe: false
Description: Apply icon type patches to AppleAHCIPort.kext to force internal disk icons for all AHCI disks.
Note: This option should be avoided whenever possible. Modern firmwares usually have compatible AHCI controllers.
8. IncreasePciBarSize
Type: plist boolean
Failsafe: false
Description: Increases 32-bit PCI bar size in IOPCIFamily from 1 to 4 GBs.
Note: This option should be avoided whenever possible. In general the necessity of this option means misconfigured or broken firmware.
9. LpicKernelPanic
Type: plist boolean
Failsafe: false
Description: Disables kernel panic on LAPIC interrupts.
10. PanicNoKextDump
Type: plist boolean
Failsafe: false
Description: Prevent kernel from printing kext dump in the panic log preventing from observing panic details. Affects 10.13 and above.
11. PowerTimeoutKernelPanic
Type: plist boolean
Failsafe: false
Description: Disables kernel panic on setPowerState timeout.

An additional security measure was added to macOS Catalina (10.15) causing kernel panic on power change timeout for Apple drivers. Sometimes it may cause issues on misconfigured hardware, notably digital audio, which sometimes fails to wake up. For debug kernels `setpowerstate_panic=0` boot argument should be used, which is otherwise equivalent to this quirk.
12. ThirdPartyDrives
Type: plist boolean
Failsafe: false
Description: Apply vendor patches to IOAHCIBlockStorage.kext to enable native features for third-party drives, such as TRIM on SSDs or hibernation support on 10.15 and newer.
Note: This option may be avoided on user preference. NVMe SSDs are compatible without the change. For AHCI SSDs on modern macOS version there is a dedicated built-in utility called `trimforce`. Starting from 10.15 this utility creates `EnableTRIM` variable in `APPLE_BOOT_VARIABLE_GUID` namespace with 01 00 00 00 value.
13. XhciPortLimit
Type: plist boolean
Failsafe: false

- **CMD+S** — single user mode.
- **CMD+S+MINUS** — disable KASLR slide, requires disabled SIP.
- **CMD+V** — verbose mode.
- **Shift** — safe mode.

7. Resolution

Type: plist string

Failsafe: Empty string

Description: Sets console output screen resolution.

- Set to **WxH@Bpp** (e.g. 1920x1080@32) or **WxH** (e.g. 1920x1080) formatted string to request custom resolution from GOP if available.
- Set to empty string not to change screen resolution.
- Set to **Max** to try to use largest available screen resolution.

On HiDPI screens **APPLE_VENDOR_VARIABLE_GUID UIScale** NVRAM variable may need to be set to **02** to enable HiDPI scaling in FileVault 2 UEFI password interface and boot screen logo. Refer to Recommended Variables section for more details.

Note: This will fail when console handle has no GOP protocol. When the firmware does not provide it, it can be added with **ProvideConsoleGop** UEFI quirk set to **true**.

8. ShowPicker

Type: plist boolean

Failsafe: false

Description: Show simple boot picker to allow boot entry selection.

9. Timeout

Type: plist integer, 32 bit

Failsafe: 0

Description: Timeout in seconds in boot picker before automatic booting of the default boot entry. Use 0 to disable timer.

10. UsePicker

Type: plist boolean

Failsafe: false

Description: Use OpenCore built-in boot picker for boot management.

UsePicker set to **false** entirely disables all boot management in OpenCore except policy enforcement. In this case a custom user interface may utilise **OcSupportPkg OcBootManagementLib** to implement a user friendly boot picker oneself. Reference example of external graphics interface is provided in **ExternalUi** test driver.

OpenCore built-in boot picker contains a set of actions chosen during the boot process. The list of supported actions is similar to Apple BDS and currently consists of the following options:

- **Default** — this is the default option, and it lets OpenCore built-in boot picker to loads the default boot option as specified in Startup Disk preference pane.
- **ShowPicker** — this option forces picker to show. Normally it can be achieved by holding **OPT** key during boot. Setting **ShowPicker** to **true** will make **ShowPicker** the default option.
- **ResetNvram** — this option performs select UEFI variable erase and is normally achieved by holding **CMD+OPT+P+R** key combination during boot. Another way to erase UEFI variables is to choose **Reset NVRAM** in the picker. This option requires **AllowNvramReset** to be set to **true**.
- **BootApple** — this options performs booting to the first found Apple operating system unless the default chosen operating system is already made by Apple. Hold **X** key to choose this option.
- **BootAppleRecovery** — this option performs booting to Apple operating system recovery. Either the one related to the default chosen operating system, or first found in case default chosen operating system is not made by Apple or has no recovery. Hold **CMD+R** key combination to choose this option.

Note: activated **KeySupport**, **UsbKbDxeAppleUsbKbDxe**, or similar driver is required for key handling to work. On many firmwares it is not possible to get all the keys function.

In addition to **OPT** OpenCore supports **Escape** key **ShowPicker**. This key exists for firmwares with PS/2 keyboards that fail to report held **OPT** key and require continual presses of **Escape** key to enter the boot menu.

File logging will create a file named `opencore-YYYY-MM-DD-HHMMSS.txt` at EFI volume root with log contents (the upper case letter sequence is replaced with date and time from the firmware). Please be warned that some file system drivers present in firmwares are not reliable, and may corrupt data when writing files through UEFI. Log is attempted to be written in the safest manner, and thus is very slow. Ensure that `DisableWatchDog` is set to `true` when you use a slow drive.

8.5 Security Properties

1. AllowNvramReset

Type: plist boolean

Failsafe: false

Description: Allow CMD+OPT+P+R handling and enable showing NVRAM Reset entry in boot picker.

2. AllowSetDefault

Type: plist boolean

Failsafe: false

Description: Allow CTRL+Enter and CTRL+Index handling to set the default boot option in boot picker.

3. AuthRestart

Type: plist boolean

Failsafe: false

Description: Enable VirtualSMC-compatible authenticated restart.

Authenticated restart is a way to reboot FileVault 2 enabled macOS without entering the password. To perform authenticated restart one can use a dedicated terminal command: `sudo fdsetup authrestart`. It is also used when installing operating system updates.

VirtualSMC performs authenticated restart by saving disk encryption key split in NVRAM and RTC, which despite being removed as soon as OpenCore starts, may be considered a security risk and thus is optional.

4. ExposeSensitiveData

Type: plist integer

Failsafe: 0x6

Description: Sensitive data exposure bitmask (sum) to operating system.

- 0x01 — Expose printable booter path as an UEFI variable.
- 0x02 — Expose OpenCore version as an UEFI variable.
- 0x04 — Expose OpenCore version in boot picker menu title.

Exposed booter path points to OpenCore.efi or its booter depending on the load order. To obtain booter path use the following command in macOS:

```
nvrnm 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:boot-path
```

To use booter path for mounting booter volume use the following command in macOS:

```
u=$(nvrnm 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:boot-path | sed 's/.*GPT,\([^,]*\),.*\/1\/'); \
if [ "$u" != "" ]; then sudo diskutil mount $u ; fi
```

To obtain OpenCore version use the following command in macOS:

```
nvrnm 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102:opencore-version
```

5. HaltLevel

Type: plist integer, 64 bit

Failsafe: 0x80000000 (DEBUG_ERROR)

Description: EDK II debug level bitmask (sum) causing CPU to halt (stop execution) after obtaining a message of HaltLevel. Possible values match DisplayLevel values.

6. RequireSignature

Type: plist boolean

Failsafe: true

Description: Require vault.sig signature file for vault.plist in OC directory.

9 NVRAM

9.1 Introduction

Has `plist dict` type and allows to set volatile UEFI variables commonly referred as NVRAM variables. Refer to `man nvram` for more details. macOS extensively uses NVRAM variables for OS — Bootloader — Firmware intercommunication, and thus supplying several NVRAM is required for proper macOS functioning.

Each NVRAM variable consists of its name, value, attributes (refer to UEFI specification), and its GUID, representing which ‘section’ NVRAM variable belongs to. macOS uses several GUIDs, including but not limited to:

- 4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14 (APPLE_VENDOR_VARIABLE_GUID)
- 7C436110-AB2A-4BBB-A880-FE41995C9F82 (APPLE_BOOT_VARIABLE_GUID)
- 8BE4DF61-93CA-11D2-AA0D-00E098032B8C (EFI_GLOBAL_VARIABLE_GUID)
- 4D1FDA02-38C7-4A6A-9CC6-4BCCA8B30102 (OC_VENDOR_VARIABLE_GUID)

Note: Some of the variables may be added by PlatformNVRAM or Generic subsections of PlatformInfo section. Please ensure that variables of this section never collide with them, as behaviour is undefined otherwise.

For proper macOS functioning it is often required to use OC_FIRMWARE_RUNTIME protocol implementation currently offered as a part of FwRuntimeServices driver. While it brings any benefits, there are certain limitations which arise depending on the use.

1. Not all tools may be aware of protected namespaces.
When RequestBootVarRouting is used Boot-prefixed variable access is restricted and protected in a separate namespace. To access the original variables tools have to be aware of OC_FIRMWARE_RUNTIME logic.
2. Assigned NVRAM variables are not always allowed to exceed 512 bytes.
This is true for Boot-prefixed variables when RequestBootVarFallback is used, and for overwriting volatile variables with non-volatile on UEFI 2.8 non-conformant firmwares.

9.2 Properties

1. Add

Type: `plist dict`

Description: Sets NVRAM variables from a map (`plist dict`) of GUIDs to a map (`plist dict`) of variable names and their values in `plist metadata` format. GUIDs must be provided in canonic string format in upper or lower case (e.g. 8BE4DF61-93CA-11D2-AA0D-00E098032B8C).

Created variables get `EFI_VARIABLE_BOOTSERVICE_ACCESS` and `EFI_VARIABLE_RUNTIME_ACCESS` attributes set. Variables will only be set if not present and not blocked. To overwrite a variable add it to `Block` section. This approach enables to provide default values till the operating system takes the lead.

Note: If `plist` key does not conform to GUID format, behaviour is undefined.

2. Block

Type: `plist dict`

Description: Removes NVRAM variables from a map (`plist dict`) of GUIDs to an array (`plist array`) of variable names in `plist string` format.

3. LegacyEnable

Type: `plist boolean`

Failsafe: `false`

Description: Enables loading of NVRAM variable file named `nvram.plist` from EFI volume root.

This file must have root `plist dictionary` type and contain two fields:

- **Version** — `plist integer`, file version, must be set to 1.
- **Add** — `plist dictionary`, equivalent to `Add` from `config.plist`.

Variable loading happens prior to `Block` (and `Add`) phases, ~~and~~. Unless LegacyOverwrite is enabled, it will not overwrite any existing variable. Variables allowed to be set must be specified in `LegacySchema`. Third-party scripts may be used to create `nvram.plist` file. An example of such script can be found in `Utilities`. The use of

third-party scripts may require `ExposeSensitiveData` set to `0x3` to provide `boot-path` variable with OpenCore EFI partition UUID.

WARNING: This feature is very dangerous as it passes unprotected data to your firmware variable services. Use it only when no hardware NVRAM implementation is provided by the firmware or it is incompatible.

4. [LegacyOverwrite](#)

[Type: plist boolean](#)

[Failsafe: false](#)

[Description: Permits overwriting firmware variables from nvram.plist.](#)

[Note: Only variables accessible from the operating system will be overwritten.](#)

5. `LegacySchema`

Type: plist dict

Description: Allows setting select NVRAM variables from a map (plist dict) of GUIDs to an array (plist array) of variable names in `plist string` format.

You can use `*` value to accept all variables for select GUID.

WARNING: Choose variables very carefully, as `nvram.plist` is not vaulted. For instance, do not put `boot-args` or `csr-active-config`, as this can bypass SIP.

6. [WriteFlash](#)

[Type: plist boolean](#)

[Failsafe: false](#)

[Description: Enables writing to flash memory for all added variables.](#)

[Note: This value is recommended to be enabled on most firmwares, but is left configurable for firmwares that may have issues with NVRAM variable storage garbage collection or alike.](#)

To read NVRAM variable value from macOS one could use `nvram` by concatenating variable GUID and name separated by `:` symbol. For example, `nvram 7C436110-AB2A-4BBB-A880-FE41995C9F82:boot-args`.

A continuously updated variable list can be found in a corresponding document: [NVRAM Variables](#).

9.3 Mandatory Variables

Warning: These variables may be added by `PlatformNVRAM` or Generic subsections of `PlatformInfo` section. Using `PlatformInfo` is the recommend way of setting these variables.

The following variables are mandatory for macOS functioning:

- `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:FirmwareFeatures`
32-bit `FirmwareFeatures`. Present on all Macs to avoid extra parsing of SMBIOS tables
- `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:FirmwareFeaturesMask`
32-bit `FirmwareFeaturesMask`. Present on all Macs to avoid extra parsing of SMBIOS tables.
- `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:MLB`
`BoardSerialNumber`. Present on newer Macs (2013+ at least) to avoid extra parsing of SMBIOS tables, especially in `boot.efi`.
- `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:ROM`
Primary network adapter MAC address or replacement value. Present on newer Macs (2013+ at least) to avoid accessing special memory region, especially in `boot.efi`.

9.4 Recommended Variables

The following variables are recommended for faster startup or other improvements:

- `7C436110-AB2A-4BBB-A880-FE41995C9F82:csr-active-config`
32-bit System Integrity Protection bitmask. Declared in XNU source code in `csr.h`.
- `4D1EDE05-38C7-4A6A-9CC6-4BCCA8B38C14:ExtendedFirmwareFeatures`
Combined `FirmwareFeatures` and `ExtendedFirmwareFeatures`. Present on newer Macs to avoid extra parsing of SMBIOS tables

- **Overwrite** — Overwrite existing gEfiSmbiosTableGuid and gEfiSmbiosTable3Guid data if it fits new size. Abort with unspecified state otherwise.
 - **Custom** — Write first SMBIOS table (gEfiSmbiosTableGuid) to gOcCustomSmbiosTableGuid to workaround firmwares overwriting SMBIOS contents at ExitBootServices. Otherwise equivalent to **Create**. Requires patching AppleSmbios.kext and AppleACPIPlatform.kext to read from another GUID: "EB9D2D31" - "EB9D2D35" (in ASCII), done automatically by CustomSMBIOSGuid quirk.
6. **Generic**
Type: plist dictionary
Optional: When Automatic is false
Description: Update all fields. This section is read only when Automatic is active.
 7. **DataHub**
Type: plist dictionary
Optional: When Automatic is true
Description: Update Data Hub fields. This section is read only when Automatic is not active.
 8. **PlatformNVRAM**
Type: plist dictionary
Optional: When Automatic is true
Description: Update platform NVRAM fields. This section is read only when Automatic is not active.
 9. **SMBIOS**
Type: plist dictionary
Optional: When Automatic is true
Description: Update SMBIOS fields. This section is read only when Automatic is not active.

10.2 Generic Properties

1. **SpoofVendor**
Type: plist boolean
Failsafe: false
Description: Sets SMBIOS vendor fields to Acidanthera.

It is dangerous to use Apple in SMBIOS vendor fields for reasons given in **SystemManufacturer** description. However, certain firmwares may not provide valid values otherwise, which could break some software.
2. SupportsCsm
Type: plist boolean
Failsafe: false
Description: Forces CSM support in FirmwareFeatures.

Without this bit it is not possible to reboot to Windows installed on a drive with EFI partition being not the first partition on the disk.

Note: This was enabled unconditionally in older OpenCore versions.
3. **SystemProductName**
Type: plist string
Failsafe: MacPro6,1
Description: Refer to SMBIOS SystemProductName.
4. **SystemSerialNumber**
Type: plist string
Failsafe: OPENCORE_SN1
Description: Refer to SMBIOS SystemSerialNumber.
5. **SystemUUID**
Type: plist string, GUID
Failsafe: OEM specified
Description: Refer to SMBIOS SystemUUID.
6. **MLB**
Type: plist string

11 UEFI

11.1 Introduction

UEFI (Unified Extensible Firmware Interface) is a specification that defines a software interface between an operating system and platform firmware. This section allows to load additional UEFI modules and/or apply tweaks for the onboard firmware. To inspect firmware contents, apply modifications and perform upgrades UEFITool and supplementary utilities can be used.

11.2 Properties

1. ConnectDrivers

Type: plist boolean

Failsafe: false

Description: Perform UEFI controller connection after driver loading. This option is useful for loading filesystem drivers, which usually follow UEFI driver model, and may not start by themselves. While effective, this option ~~is not necessary with e.g. APFS loader driver~~ may not be necessary for drivers performing automatic connection, and may slightly slowdown the boot.

2. Drivers

Type: plist array

Failsafe: None

Description: Load selected drivers from `OC/Drivers` directory.

Designed to be filled with string filenames meant to be loaded as UEFI drivers. Depending on the firmware a different set of drivers may be required. Loading an incompatible driver may lead your system to unbootable state or even cause permanent firmware damage. Some of the known drivers include:

- **ApfsDriverLoader** — APFS file system bootstrap driver adding the support of embedded APFS drivers in bootable APFS containers in UEFI firmwares.
- **FwRuntimeServices** — `OC_FIRMWARE_RUNTIME` protocol implementation that increases the security of OpenCore and Lilu by supporting read-only and write-only NVRAM variables. Some quirks, like **RequestBootVarRouting**, require this driver for proper function. Due to the nature of being a runtime driver, i.e. functioning in parallel with the target operating system, it cannot be implemented within OpenCore itself, but is bundled with OpenCore releases.
- **EnhancedFatDxe** — FAT filesystem driver from **FatPkg**. This driver is embedded in all UEFI firmwares, and cannot be used from OpenCore. It is known that multiple firmwares have a bug in their FAT support implementation, which leads to corrupted filesystems on write attempt. Embedding this driver within the firmware may be required in case writing to EFI partition is needed during the boot process.
- **NvmExpressDxe** — NVMe support driver from **MdeModulePkg**. This driver is included in most firmwares starting with Broadwell generation. For Haswell and earlier embedding it within the firmware may be more favourable in case a NVMe SSD drive is installed.
- **AppleUsbKbDxe** — USB keyboard driver adding the support of **AppleKeyMapAggregator** protocols on top of a custom USB keyboard driver implementation. This is an alternative to builtin **KeySupport**, which may work better or worse depending on the firmware.
- ~~— UEFI SMC driver, required for proper FileVault 2 functionality and potentially other macOS specifics. An alternative, named **SMCHelper**, is not compatible with **VirtualSmc** and OpenCore, which is unaware of its specific interfaces. In case **FakeSMC** kernel extension is used, manual NVRAM variable addition may be needed and **VirtualSmc** driver should still be used.~~
- **VBoxHfs** — HFS file system driver with bless support. This driver is an alternative to a closed source **HFSPlus** driver commonly found in Apple firmwares. While it is feature complete, it is approximately 3 times slower and is yet to undergo a security audit.
- **XhciDxe** — XHCI USB controller support driver from **MdeModulePkg**. This driver is included in most firmwares starting with Sandy Bridge generation. For earlier firmwares or legacy systems it may be used to support external USB 3.0 PCI cards.

To compile the drivers from UDK (EDK II) use the same command you do normally use for OpenCore compilation, but choose a corresponding package:

```
git clone https://github.com/acidanthera/audk UDK
```

```
cd UDK
source edksetup.sh
make -C BaseTools
build -a X64 -b RELEASE -t XCODE5 -p FatPkg/FatPkg.dsc
build -a X64 -b RELEASE -t XCODE5 -p MdeModulePkg/MdeModulePkg.dsc
```

3. Input

Type: plist dict

Failsafe: None

Description: Apply individual settings designed for input (keyboard and mouse) in Input Properties section below.

4. Protocols

Type: plist dict

Failsafe: None

Description: Force builtin versions of select protocols described in Protocols Properties section below.

Note: all protocol instances are installed prior to driver loading.

5. Quirks

Type: plist dict

Failsafe: None

Description: Apply individual firmware quirks described in Quirks Properties section below.

11.3 Input Properties

1. KeyForgetThreshold

Type: plist integer

Failsafe: 0

Description: Remove key unless it was submitted during this timeout in milliseconds.

AppleKeyMapAggregator protocol is supposed to contain a fixed length buffer of currently pressed keys. However, the majority of the drivers only report key presses as interrupts and pressing and holding the key on the keyboard results in subsequent submissions of this key with some defined time interval. As a result we use a timeout to remove once pressed keys from the buffer once the timeout expires and no new submission of this key happened.

This option allows to set this timeout based on your platform. The recommended value that works on the majority of the platforms is 5 milliseconds. For reference, holding one key on VMware will repeat it roughly every 2 milliseconds and the same value for APTIO V is 3-4 milliseconds. Thus it is possible to set a slightly lower value on faster platforms and slightly higher value on slower platforms for more responsive input.

2. KeyMergeThreshold

Type: plist integer

Failsafe: 0

Description: Assume simultaneous combination for keys submitted within this timeout in milliseconds.

Similarly to **KeyForgetThreshold**, this option works around the sequential nature of key submission. To be able to recognise simultaneously pressed keys in the situation when all keys arrive sequentially, we are required to set a timeout within which we assume the keys were pressed together.

Holding multiple keys results in reports every 2 and 1 milliseconds for VMware and APTIO V respectively. Pressing keys one after the other results in delays of at least 6 and 10 milliseconds for the same platforms. The recommended value for this option is 2 milliseconds, but it may be decreased for faster platforms and increased for slower.

3. KeySupport

Type: plist boolean

Failsafe: false

Description: Enable internal keyboard input translation to AppleKeyMapAggregator protocol.

This option activates the internal keyboard interceptor driver, based on AppleGenericInput aka (AptioInputFix), to fill AppleKeyMapAggregator database for input functioning. In case a separate driver is used, such as ~~UsbKbDxe~~AppleUsbKbDxe, this option should never be enabled.

4. **AppleKeyMap**
Type: plist boolean
Failsafe: false
Description: Reinstalls Apple Key Map protocols with builtin versions.
5. [AppleSmcIo](#)
[Type:](#) plist boolean
[Failsafe:](#) false
[Description:](#) Reinstalls Apple SMC I/O protocol with a builtin version.
[This protocol replaces legacy VirtualSmc EFI driver, and is compatible with any SMC kernel extension. However, in case FakeSMC kernel extension is used, manual NVRAM key variable addition may be needed.](#)
6. **AppleUserInterfaceTheme**
Type: plist boolean
Failsafe: false
Description: Reinstalls Apple User Interface Theme protocol with a builtin version.
7. **ConsoleControl**
Type: plist boolean
Failsafe: false
Description: Replaces Console Control protocol with a builtin version.

macOS bootloader requires console control protocol for text output, which some firmwares miss. This option is required to be set when the protocol is already available in the firmware, and other console control options are used, such as `IgnoreTextInGraphics`, `SanitiseClearScreen`, and sometimes `ConsoleBehaviourOs` with `ConsoleBehaviourUi`).
8. **DataHub**
Type: plist boolean
Failsafe: false
Description: Reinstalls Data Hub protocol with a builtin version. This will drop all previous properties if the protocol was already installed.
9. **DeviceProperties**
Type: plist boolean
Failsafe: false
Description: Reinstalls Device Property protocol with a builtin version. This will drop all previous properties if it was already installed. This may be used to ensure full compatibility on VMs or legacy Macs.
10. **FirmwareVolume**
Type: plist boolean
Failsafe: false
Description: Forcibly wraps Firmware Volume protocols or installs new to support custom cursor images for File Vault 2. Should be set to `true` to ensure File Vault 2 compatibility on everything but VMs and legacy Macs.
11. **HashServices**
Type: plist boolean
Failsafe: false
Description: Forcibly reinstalls Hash Services protocols with builtin versions. Should be set to `true` to ensure File Vault 2 compatibility on platforms providing broken SHA-1 hashing. Can be diagnosed by invalid cursor size with `UIScale` set to 02, in general platforms prior to APTIO V (Haswell and older) are affected.
12. [OSInfo](#)
[Type:](#) plist boolean
[Failsafe:](#) false
[Description:](#) Forcibly reinstalls OS Info protocol with builtin versions. This protocol is generally used to receive notifications from macOS bootloader, by the firmware or by other applications.
13. **UnicodeCollation**
Type: plist boolean
Failsafe: false